MODELLING AND OPTIMISATION OF MANUFACTURING SYSTEMS: PETRI NETS AND SIMULATED ANNEALING

A. Zimmermann*, D. Rodriguez†, M. Silva†

Techn. Univ. of Berlin, Franklinstr. 28/29, Sekr. FR 2-2, 10587 Berlin Germany
Fax: +49-30-31421116 e-mail: azi@cs.tu-berlin.de
† Univ. of Zaragoza, c/Maria de Luna 3, Zaragoza 50015 Spain
Fax: +34-976-762111 e-mail: dierodri, silva@posta.unizar.es

Keywords : Manufacturing systems, Modelling, Optimisation, Petri nets, Performance Evaluation.

Abstract

Optimisation during the design of large manufacturing systems is a key issue. An adequate modelling formalism to express the intricate interleaving of competition and cooperation relationships is needed first. Moreover, robust and efficient optimisation techniques are necessary. This paper presents an integrated tool for the automated optimisation of DEDS, with application to manufacturing systems. After overviewing optimisation problems in Manufacturing Systems, it presents the integration of existing tools for the modelling and evaluation with Petri nets and a general-purpose optimisation package based on Simulated Annealing. Two application examples show the benefits of the proposed technique.

1 Introduction

The design of modern manufacturing systems is a complex task. High investments necessitate to make sure that the planned system will fulfil the requirements. Methods and computer tools for the modelling and performance evaluation and optimisation of manufacturing systems are therefore important.

Discrete event dynamic systems correspond to a view of systems where the state space is discrete (i.e. states are countable) and state changes are driven by (external or internal) events. The DEDS view of systems has been present in systems theory for a long time. Currently it gains importance by the ever increasing development of computer based technologies. Manufacturing systems design and operation is one of the technological fields where the DEDS view is widely used [18].

The complexity of the behaviour of DEDS requires formal means to model systems. Moreover, the necessity to optimise their design or operation calls for new optimisation meta-heuristics. In this paper we consider Petri nets for the modelling and analysis of manufacturing systems, while design optimisation is approached using Simulated Annealing.

The use of Petri nets allows to model systems with intricate interleaving of cooperation and competition, thanks to the ability of nets to model conflicts and synchronisations. Provided with appropriate interpreted extensions, PNs lead to different formalisms useful in the different phases of the life-cycle of the system under design or operation, thus constituting a formal modelling *paradigm* [17].

Petri net performance models (PNPM) are obtained through interpreted extension of autonomous models in which a time duration is associated with transitions, and routing policies are defined to solve conflicts. The complex interleaving of choices and synchronisations in manufacturing systems may lead to systems where paradoxical behaviours are exhibited. For example, increasing the number of resources (i.e. tokens in the net model) can lead to a dead-locked system, and replacing a machine for a faster one can decrease the global productivity. It is clear that formal techniques and computer tools are required for the design and optimisation of complex manufacturing systems. In this paper we adopt an evaluative meta-heuristic (Simulated Annealing) to solve the different optimisation problems.

2 Design Problems in Manufacturing System

This section identifies some of the design problems that frequently occur in the design of a manufacturing system. Problems usually involve the selection of one out of sev-

*This work was performed while A. Zimmermann was visiting the University of Zaragoza

†This work has been partially developed within the project TAP. 98-0679 of the Spanish CICYT

eral options (e.g. a machine selection), or a dimensioning. Numerical values can be discrete (e.g. the size of a buffer) or continuous (e.g. a production mix). Typical design problems are:

Machine selection where reliable machine selection problems (high or low speed machines with their costs) and failure problems (machines with different failure and repair times) are considered. It can lead to *line balancing* problems.

Production mix problems appears when the relative production rates of several part types should be optimised (eventually under production and marketing constraints). The question is to find an "optimal" percentage of each product type. Parameters to optimise are non negative real values.

Buffer allocation problems are very common in Manufacturing Systems. Typical examples are: the buffer allocation problem where decisions must be taken in order to determine *if* and *where* buffers are needed; buffer *sizing*, which is a numerical problem where we have to choose the optimal dimension of the buffer (the selection of the number of *pallets* or automated guided vehicles are mathematically similar problems).

Material Handling System refers to the different problems arising in the transport of parts. Different types of problems can be distinguished here, the first type being the selection of the type of MHS elements (AGV, conveyors, ...) and the topology. Finally, the number of conveyors, automated guided vehicles, etc. and their speed has to be selected.

In practice, mixtures of the above mentioned problems apply to optimisation of manufacturing systems. The example in Section 4.1 considers a typical buffer allocation problem for a two-parts production line, where the size of the buffers should be optimised. In Section 4.2 the example explores simultaneously some different production strategies, buffer sizing and machine selection. Typically optimisations deals with complex non-linear evaluative models in high-dimensional search spaces.

3 Design optimisation

During the design of a manufacturing system the basic task is to choose between design options, as explained in Section 2, such that an optimal behaviour is achieved. Many different types of optimisation techniques are available. Some optimisation techniques are *generative* (given a set of criteria and dynamic constraints generate a set of decisions) while others are *evaluative* (given a set of decisions evaluate the performance of the system). Among the first one are those problems that allow to fit well-known mathematical programming templates like linear- (mixed, integer, ...) programming problems. For many design optimisation problems, like buffer size or production mix optimisation, evaluative techniques are frequently used in an iterative schema with some alternatives generation strategy.

Discrete optimisation problems in Manufacturing Systems are usually NP-hard. For real-life problems it is not possible to analyse the full parameter space. Therefore search techniques have been developed that in general do not guarantee to find the global optimum, but often lead to a "good" (or just "acceptable") solution. Modern optimisation techniques approach the problem through some *meta-heuristics*, e.g. *tabu search* [16], *genetic algorithms* [16, 8], and *simulated annealing* [2, 13]. The latter is used here and shortly explained in the following section.

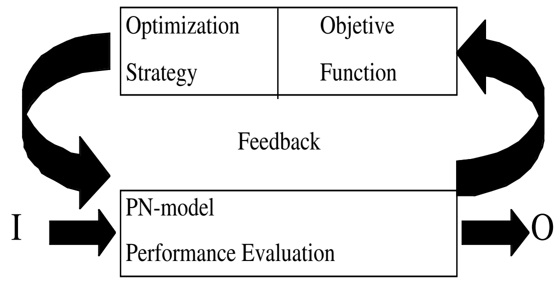

Figure 1: Evaluative Techniques: Iterative Optimisation

As can be seen in Figure 1, the optimisation process will be an iterative process between the PNPM and the optimisation strategy (the strategy is in this case simulated annealing but we can consider others, as mentioned before). The performance measures can be obtained using *analytical* techniques (e.g. product form solutions, if possible), *numerical* techniques (Markov-chain, brute force or net-driven generated) or *simulation* techniques. In most practical cases, expensive model evaluations are required. A broad perspective of PNPM performance evaluation techniques is contained in [3].

Manufacturing systems are set up in order to make profit from producing and selling parts. Therefore, a *profit function* has to be specified and later maximised by the optimisation. Typical profit functions should consider the money earned from selling finished parts minus the costs arising in the production process. The price of raw parts, the money spent for work-in-process, machine and transport systems amortisation, and constant as well as utilisation dependent costs are examples. All of them can be determined by a performance analysis of a model.

3.1 Simulated Annealing

Simulated Annealing is a widely applicable technique for discrete optimisation. It proceeds by simulating the change of energy of the system until "convergence" to a *frozen state*. This method can be applied to a local search problem where we are moving continuously from the current solutions to neighbouring solutions. It is based on a simple formula based on the increase of energy magnitude δE which is given by:

$$p(\delta E) = exp(-\delta t/kt)$$

where k is a constant.

A major advantage of simulated annealing is that it can accept a worse solution temporarily. The algorithm is therefore able to leave local optima. A generic simulated annealing algorithm (see [2] for details) works as follows: Let S be the solution space, the objective function to be minimised f, and the neighbourhood structure is denoted with N.

Select an initial solution s_0;
Select an initial temperature $t_0 > 0$;
Select a temperature reduction function α;
repeat
    repeat
        Randomly select $s \in N(s_0)$;
        $\delta = f(s) - f(s_0)$;
        if $\delta < 0$
            then $s_0 = s$;
            else generate random x in $(0, 1)$;
        if $x < exp(-\delta/t)$ **then** $s_0 = s$;
    until $iteration - count = nrep$
    Set $t = \alpha(t_0)$;
until exit condition = true
s_0 is the found solution

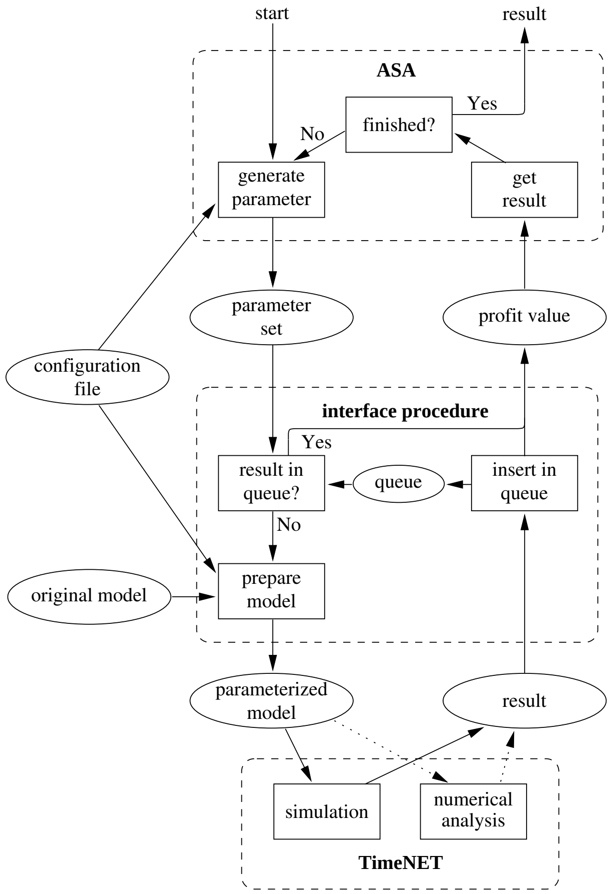

Figure 2: Interaction of ASA and TimeNET

3.2 An Integrated Software Tool for Manufacturing System Optimisation

We decided to use the software package ASA (Adaptive Simulated Annealing) [12, 13], which implements the simulated annealing technique for optimising multivariate nonlinear complex systems.

For modelling and automated evaluation of manufacturing systems with Petri nets an additional computer tool was necessary. We used the tool TimeNET [10] (timed net evaluation tool). TimeNET is a software package for the modelling and evaluation of stochastic Petri nets in which the firing times of the transitions may be exponentially distributed, deterministic, or more generally distributed. Besides numerical analysis techniques, TimeNET also supplies a *simulation* component for non-Markovian Petri nets [14]. This module provides statistical techniques for a reliable variance estimation and derivation of valid confidence intervals. Simulation runs can be executed in parallel on a cluster of workstations. Figure 2 shows in an informal way the interaction of ASA and TimeNET during an optimisation run.

The computational effort for computing the value of the cost function from a parameter set is very small for usual applications of the ASA package. This is not the case here, because every cost function is computed through a TimeNET call. Such an analysis or simulation can take some minutes to complete, depending on the model size or the confidence interval. During the optimisation, the same parameter sets (or only slightly differing ones) are often generated. It is therefore very important for efficiency to avoid re-computations. Every result is thus stored in a table, called here *queue* together with its corresponding parameter set by the interface procedure.

Before an optimisation can be started, the *original model* and a *configuration file* have to be specified. The configuration file contains the objective function and the parameters to consider in the optimisation. The model is constructed using the TimeNET graphical user interface. It must contain the definition of a cost or profit function as a performance measure. ASA calls its user-defined cost function, which is now the interface procedure to TimeNET, with a parameter set. The parameters given to the interface procedure are first checked whether they have already been computed. If a result is found, it is directly returned to ASA without calling TimeNET. Experience shows that the presence of the queue minimises the overall computational effort substantially.

In the case that the parameter set has not been evaluated, the interface procedure prepares a parameterised model from the original Petri net model by substituting the actual parameter values in the model description. A performance evaluation component of TimeNET is called afterwards. The resulting file with the computed value of the profit or cost function is read by the interface procedure after the TimeNET process has finished. The new value is stored in the queue together with the parameter set and afterwards returned to the ASA optimiser. In the next step ASA tests whether convergence is reached and exits with the final optimisation result. A new parameter set is generated otherwise and a new iteration begins.

4 Examples

The purpose of this section is to present two characteristic examples. The variables involved in the optimisation problems are discrete (buffer size) or continuous (speed of machines).

4.1 Optimising a Production Line

A production line is modelled with the Petri net shown in Figure 3. In this case we have clearly *competition* relationships (sharing of resources) together with some important *variability* introduced by failures and repairs that machine ($\mathcal{M}_2$) suffers. There exists no explicit cooperation at the functional level (it appears by additivity of profits concerning the two different part types). The optimisation problem is clearly a discrete one, even though a production mix (continuous variable) is obtained as a result.

The manufacturing system consists of three machines ($\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$), which can process one part at a time each. Between the machines there are two intermediate buffers $\mathcal{B}_1$ and $\mathcal{B}_2$. Processing times are deterministic (filled rectangles depict deterministic transitions), while transport time to and from the buffers are modelled with exponential distributions. Two different types of products, named A and B, are produced. The upper and lower part of the model specify the different processing steps that correspond to the two products. Places in those two lines model processing states of a work piece or an intermediate buffer, while the places in between correspond to buffer and machine capacities.

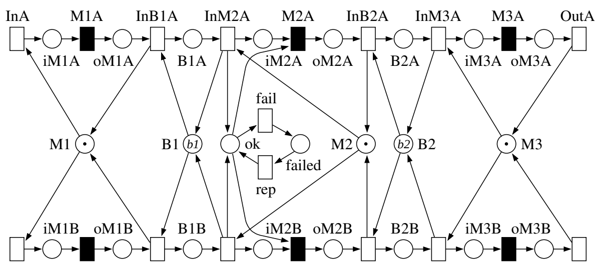

Figure 3: Production Line with three machines

Machine $\mathcal{M}_1$ is loaded by firing transition `InA`, which is also the system input for products of type A. The firing of transition `M1A` corresponds to the completion of the first processing task in machine $\mathcal{M}_1$. The third machine is similar to the first one, while $\mathcal{M}_2$ is subject to failures (transition `fail` fires) and repairs (transition `rep` fires). Machining times are fixed and thus modelled using transitions with deterministic firing times. Transport times are exponentially distributed.

The buffers $\mathcal{B}_1$ and $\mathcal{B}_2$ act as intermediate storage places where work pieces (types A or B) are kept until the following machine is idle. This is specially useful if machines might fail. Without buffers, one failed machine starves or blocks all other machines after a short period of time, which decreases the overall throughput of the system. Like in electrical circuits, buffers (capacities) try to filter the high frequency variations in the behaviour of the system, decoupling up and down streams.

The question is then: where should buffers be put and how should their capacity be adjusted? This will of course depend on the speed of the machines and their failure and repair behaviour. For the following experiments, the second machine is the failing one (as shown in Figure 3), while the last machine is the slowest of all three. The aim of the optimisation is then to maximise the profit of the production line, while the number of buffer places of buffers one and two are changed in the range from 1 to 19.

The second important step after defining a model is to specify a profit function. The profit function will compute the profit per day. For this model it considers the profit per part. As there is one raw part to be bought for every sold one, we only need the difference between the two prices. Assume this profit per part to be 70 for parts of type A, and 60 for those of type B. Then we need to compute the throughput of parts of both types from the model.

The time horizon of the production line is set to one year. We assume an investment of 500000 plus 4000 for each buffer position. The amount for buffers includes other costs that depend on the number of buffers, such as maintenance. Constant costs are set to 1500 per day.

For an assessment of the optimisation results and the efficiency of the used algorithm we display the profit function for the full parameter space (see Figure 4). The number of places of the two buffers were varied in the range from 1 to 19 with a step size of 2.

Figure 4 shows a surface plot of the results. The profit function result is in the range between 950 and 1270. In this case, it is evident that the influence of $\mathcal{B}_2$, which is between machines $\mathcal{M}_2$ (failing machine) and $\mathcal{M}_3$ (bottleneck machine), is greater than the one of $\mathcal{B}_1$, which is between machine $\mathcal{M}_1$ and $\mathcal{M}_2$ (failing machine).

Performance evaluation is done through simulation. The confidence level is set to 99 percent, and the maximum allowed error is one percent. The calculations have been done in parallel on a cluster of 10 UltraSparc workstations. One simulation run took approximately one minute.

The step size for the buffer places is set to one for a more exact optimisation. An automatic optimisation with the ASA/TimeNET tool described in Section 3.2 then finds

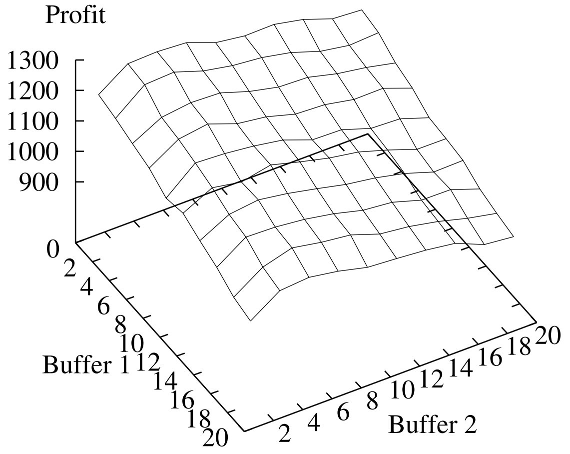

Figure 4: Profit function versus buffer places

an optimal solution at ($\mathcal{B}_1 = 1$, $\mathcal{B}_2 = 4$), for which the profit is 1280. The smaller step size means that there are $19 * 19 = 361$ possible parameter sets. During the optimisation, only 74 different sets are analysed (thus, 20 percent of the parameter space is directly investigated here). Out if the 692 subsequent profit function calls of the ASA algorithm, 618 could be answered from the queue of previous computations, which means a "queue hit rate" of 89 percent. This demonstrates the importance of this filtering data structure for the overall efficiency of the optimisation.

If the failing machine is not the bottleneck, as a rule of thumb we can say that *the buffers have to be placed between bottleneck and failing machine.* This makes sense because the bottleneck machine should never be starved or blocked, for which the probability can be reduced by putting buffer place between it and the failing machine. If those two machines are not subsequent ones in the production line, it does not matter where the buffer place is put between them.

4.2 Optimising an Assembly Line

An assembly line with five machines producing one final part is considered in this second example. Three different parts named A, B and C and additional smaller parts have to be assembled for one final product. Resource sharing takes place because some of the five machines are responsible for more than one assembly operation. The assembly sequence is given as follows: Parts of type A arrive at machine 1, where an additional unspecified part is assembled. They are transported to machine 3, where a second part of this type is added. Raw parts of type B arrive at machine 2. After an assembly operation there, and at the following machine 1, they are put together with a part of type A by machine 4. Parts of type C pass machines 3, 1 and 4, before they are assembled with the part containing A and B by machine 5. Therefore, at the functional level this example shows an important interleaving among cooperation (assembly) and competition (resource sharing) relationships. Failures and repairs are not considered here.

The task of the planned assembly system is to fulfil customer demands. We assume that only parts ordered by a customer are sold. Thus, customer waiting times for finished parts are more important than the actual throughput. The problem is to find a balance between work in process and small waiting times.

Three different control policies are analysed to cover the range of possible solutions. The first strategy for minimising the waiting time is to produce parts until the buffers are full. We refer to this as the "push" strategy. The opposite is the "on demand" strategy, where an assembly operation is started only after an order is received. Another solution, that is considered as a balance between the other two, is the "kanban" strategy, which belongs to the group of "pull"-type policies. There are intermediate buffers after groups of processing steps, which lead to smaller waiting times.

The second important parameter of the planned system is the number of parts in the system. This is controlled by allocating a certain amount of buffer places or by selecting the number of kanban cards. Finally, a machine selection problem is also considered. We assume that there are three different options for machine 1. Each of them has a different processing speed (a faster machine is more expensive).

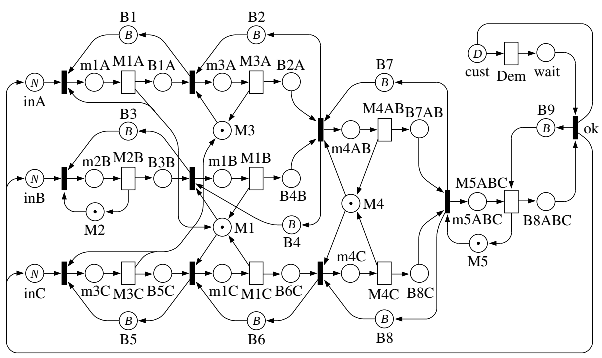

Figure 5: Assembly line (5 machs.): a push strategy

Figure 5 shows a Petri net model of the system with push strategy. The three raw parts A, B, and C arrive in the places `inA`, `inB`, and `inC`. Machines are modelled by a resource place with the name of the machine, e.g. `M1`, and a sequence of an immediate transition, an operation place (like `m1A`), and a transition (like `M1A`). There are nine intermediate buffers (named `B1`...`B9`), which all have a uniform capacity of B parts (B is the buffer size and can be used to control the maximum work in process).

The customers are modelled with the upper right part.

There are D customers which can either be inactive (token in place `cust`) or waiting for an order to be finished (token in place `wait`). The firing time of transition `Dem` sets the expected time between customer orders. During the operation of the system, the output buffer (place `B8ABC`) is filled with finished products. Transition `ok` fires when an order arrives and a complete part is available in this buffer.

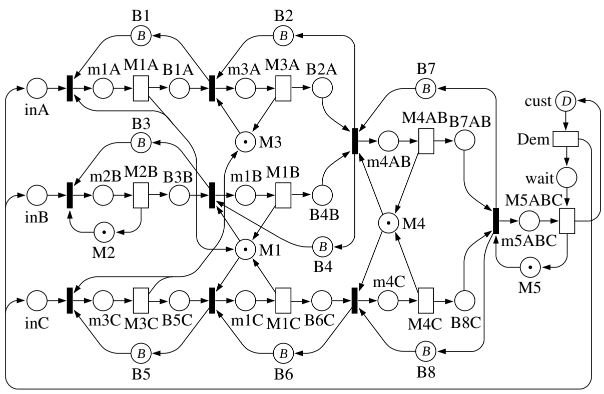

Figure 6: Assembly line with "on demand" strategy

Figure 6 shows the model of the system with a "on demand" strategy. It is very similar to the previous model depicted in Figure 5. The difference is the way how customer demands are treated. Every order (transition `Dem` fires) starts the production of a new part by adding tokens to places `inA`, `inB`, and `inC`. The system is idle and empty as long as there are no orders. The firing of transition `M5ABC` models the final assembly step of a part, which is instantaneously delivered while the corresponding customer becomes inactive again (place `cust`). Again, the amount of work in process can be controlled by adjusting the buffer size $\mathcal{B}$.

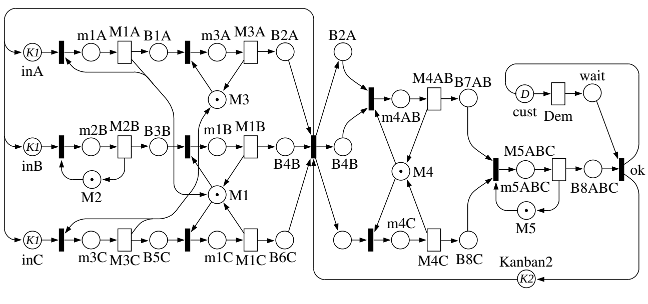

Figure 7: Assembly line with Kanban strategy

The model of the assembly system with a kanban strategy is shown in Figure 7. There are now two stages of assembly operations, which are connected by a central immediate transition. The operation is controlled by kanban cards. For the first stage their number is given by $K1$ and for the second by $K2$. Each assembly stage produces parts and stores them in its output buffer as long as there are kanban cards available. The number of cards therefore directly corresponds to the maximum number of parts in the stage, and thus the work in process. $K1$ and $K2$ are parameters for the optimisation. Additionally, the firing delay of transitions `M1A`, `M1B`, and `M1C` is changed according to the selected machine 1 as for the other two models.

For the calculation of the profit per part we assume here that customers pay less if they have to wait longer for an ordered part. A finished part without waiting time costs 1000, and the raw parts for one product cost 400. The waiting time is included in the profit function. Additionally, investment amortisation is considered in the profit function, depending on the selected machine 1. Work in process is computed from the mean number of parts in the system and is included in the profit function as well as constant working costs.

The profit function for the range of possible parameter sets has been evaluated in order to test the optimisation algorithm. Figure 8 shows the profit function for the kanban strategy versus the numbers of kanban cards in the two assembly stages. There is one mesh of plot data for each of the three possible processing delays of machine 1. The profit is small or even negative for a slow machine 1. The other two meshes differ substantially only for small numbers of kanban cards, where the fastest machine 1 is better. Profit results of about 24000 are achieved using machine 1 with delay 5, and kanban card numbers of 4 and higher. A machine 1 with delay 1 results in about 5 percent less profit.

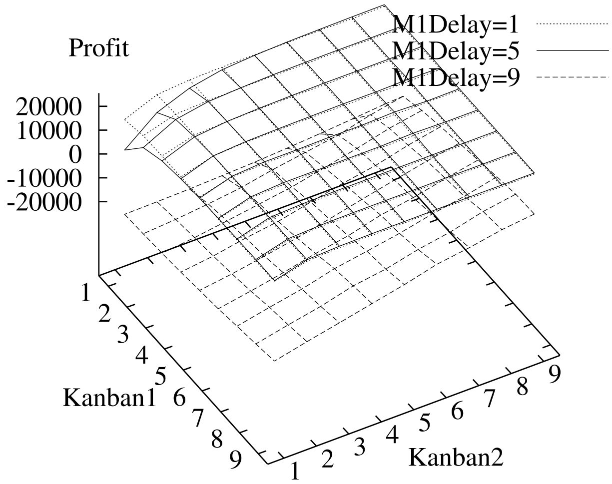

Figure 8: Assembly line with kanban strategy

The surface plots for the other two strategies are omitted here. For the push strategy (Fig. 5), the best results of about *23000* are achieved for $M1Delay = 5$ and a number of buffer spaces less than 7. If the less expensive machine with delay 9 is chosen, the profit drops below zero for buffer sizes bigger than 5. For the on demand strategy the profit is negative for any one of the parameter sets. This strategy is clearly not suitable for our task, because production orders start under customers demands (too much delay in the feedback cooperation).

During the optimisation, one special type of parameter selects between the three available models. The automatic optimisation finds a profit result of 24300 by selecting the kanban strategy, $K1 = 4, K2 = 7$ and $M1Delay = 5$. In this case 95 out of 729 theoretically possible parameter sets are evaluated during the search for an optimum. Only 13 percent of the optimiser calls to the profit function had to be answered by starting a TimeNET simulation, while the results of the rest had already been stored in the queue due to previous calculations (i.e. the hit ratio in the queue is of 87 percent).

5 Concluding remarks

Techniques and tools for the modelling, performance evaluation, and optimisation of DEDS have been considered. Performance evaluation methods based on Petri nets and a general-purpose optimisation algorithm are integrated into a single tool.

Manufacturing systems are adequately modelled with Petri nets. Steady-state analysis methods or simulation compute performance measures like the throughput, work in process and others subsequently. An adaptive simulated annealing meta-heuristic is used for the studied type of problems. The ASA software package implementing this technique [12, 13] has been adapted for the optimisation of the Petri net models. Profit function values are computed using the performance evaluation algorithms of TimeNET [10]. The integrated tool support shows its applicability using two examples where different relationships and objectives are evaluated.

Of particular interest is the already computed *queue*, avoiding to recompute evaluation results. Its hit ratio goes up to 85-90 percent, what reduces the computational burden by one order of magnitude in the many considered examples.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets" *Wiley* (1995)

[2] E. Aarts, J. Korst, "Simulated Annealing and Bolzmann Machines" *Wiley* (1989)

[3] G. Balbo, M. Silva (ed.), "Performance Models for Discrete Events Systems with Synchronizations: Formalism and Analysis Techniques" (Vols. I and II) *MATCH Summer School, Jaca* (1998)

[4] J. Browne, C. Heavey, H.T. Papadopoulos, "Queueing theory of Manufacturing Systems. Analysis and Design" *Chapman-Hall* (1993)

[5] G. Ciardo, R. German, C. Lindemann, "A Characterization of the Stochastic Process Underlying a Stochastic Petri Net" *IEEE Transactions on Software Engineering*, **20**, pp. 506–515 (1994)

[6] Y. Dallery, Z. Liu, and D. Towsley, "Equivalence, Reversibility, Symmetry and Concavity Properties in Fork-Join Queueing Networks with Blocking" *Journal of the ACM*, **41**, pp. 903–942 (1994)

[7] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, F.B. Vernadat, "Practice of Petri Nets in Manufacturing" *Chapman-Hall* (1993)

[8] M. Gen, R. Cheng, "Genetic Algorithms and Engineering Design" *Wiley* (1997)

[9] S. B. Gershwin, "Manufacturing Systems Engineering" *Prentice Hall* (1994)

[10] R. German, C. Kelling, A. Zimmermann, G. Hommel, "TimeNET - A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets" *Journal of Performance Evaluation*, **24**, pp. 69–87 (1995)

[11] W. J. Gordon, and G. F. Newell, "Closed Queueing Systems with Exponential Servers" *Operations Research*, **15**, pp. 254–265 (1967)

[12] L. Ingber, "Very fast simulated re-annealing". *Journal of Mathematical Computer Modelling*, **12 No. 8**, pp. 967–973 (1989)

[13] L. Ingber, "Adaptive simulated annealing (ASA): Lessons learned" *Journal of Control and Cybernetics*, **25 No. 1**, pp. 33–54 (1996)

[14] Ch. Kelling, "TimeNET$_{SIM}$ – a Parallel Simulator for Stochastic Petri Nets" *Proc. 28th Annual Simulation Symposium, Phoenix, AZ, USA*, pp. 250–258 (1995)

[15] J.M. Proth, X.L. Xie, "Petri Nets. A tool for design and Management of Manufacturing Systems" *Prentice Hall* (1996)

[16] C.L. Reeves, "Modern Heuristic techniques for Combinatorial Problems" *Wiley* (1993)

[17] M. Silva, E. Teruel, "A systems theory perspective of discrete event dynamic systems: the Petri net paradigm" *Symposium on Discrete Events and Manufacturing Systems, CESA-IMACS Multiconference, Lille, France* pp. 1–12 (1996)

[18] M. Silva, E. Teruel, "Petri Nets for the Design and Operation of Manufacturing Systems" *European Journal of Control*, **3 No. 3**, pp. 182–199 (1997)